Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/JP2005/020000

International filing date: 31 October 2005 (31.10.2005)

Document type: Certified copy of priority document

Document details: Country/Office: JP
Number: 2004-350702
Filing date: 03 December 2004 (03.12.2004)

Date of receipt at the International Bureau: 03 January 2006 (03.01.2006)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出　願　年　月　日
Date of Application:　　２００４年１２月　３日

出　願　番　号
Application Number:　　特願２００４－３５０７０２

パリ条約による外国への出願に用いる優先権の主張の基礎となる出願の国コードと出願番号
The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is　　ＪＰ２００４－３５０７０２

出　願　人
Applicant(s):　　株式会社ソニー・コンピュータエンタテインメント

２００５年１２月１４日

特許庁長官
Commissioner,
Japan Patent Office　　中嶋　誠

【書類名】　特許願
【整理番号】　SCEI04116
【提出日】　平成16年12月 3日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 12/00
【発明者】
　【住所又は居所】　東京都港区南青山２丁目６番２１号　株式会社ソニー・コンピュータエンタテインメント内
　【氏名】　野田　慎治
【発明者】
　【住所又は居所】　東京都港区南青山２丁目６番２１号　株式会社ソニー・コンピュータエンタテインメント内
　【氏名】　河野　健
【特許出願人】
　【識別番号】　395015319
　【氏名又は名称】　株式会社ソニー・コンピュータエンタテインメント
【代理人】
　【識別番号】　100105924
　【弁理士】
　【氏名又は名称】　森下　賢樹
　【電話番号】　03-3461-3687
【手数料の表示】
　【予納台帳番号】　091329
　【納付金額】　16,000円
【提出物件の目録】
　【物件名】　特許請求の範囲　1
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1

【書類名】特許請求の範囲
【請求項１】
第１のプロセッサと第２のプロセッサを含み、
第１のプロセッサは、実行中のメインルーチンにおいて所定のコール命令を実行したとき第２のプロセッサに対して割込を発生する割込発生部を備え、
第２のプロセッサは、割込発生部から割込を受けたとき、前記コール命令によって呼び出されるサブルーチンの処理が完了したときメインルーチンへ戻るための戻りアドレスを所定のメモリ領域に退避するアドレス退避部を備えることを特徴とするマルチプロセッサシステム。
【請求項２】
請求項１に記載のシステムにおいて、
前記割込発生部は、サブルーチンにおいて所定のリターン命令を実行したとき第２のプロセッサに対して再度割込を発生し、
第２のプロセッサは、再度発生された割込を受けたとき、前記戻りアドレスを第１のプロセッサへ通知するアドレス通知部をさらに備えることを特徴とするマルチプロセッサシステム。
【請求項３】
請求項２に記載のシステムにおいて、第１のプロセッサは、命令をフェッチするフェッチャーを備え、このフェッチャーによるアクセスの対象アドレスとして前記戻りアドレスが設定されることを特徴とするマルチプロセッサシステム。
【請求項４】
第１のプロセッサと第２のプロセッサを含み、
第１のプロセッサは、所定のコール命令またはジャンプ命令を実行したとき第２のプロセッサに対して割込を発生する割込発生部を備え、
第２のプロセッサは、
第１のプロセッサから割込を受けたとき、前記コール命令またはジャンプ命令と、その後に置かれる実行停止命令のフォーマット中に分割して格納されたコール先アドレスまたはジャンプ先アドレスを抽出するアドレス抽出部と、
取得したコール先アドレスまたはジャンプ先アドレスを第１のプロセッサに通知するアドレス通知部と、
を備えることを特徴とするマルチプロセッサシステム。
【請求項５】
請求項４に記載のシステムにおいて、第１のプロセッサは、命令をフェッチするフェッチャーを備え、このフェッチャーによるアクセスの対象アドレスとして前記コール先アドレスまたはジャンプ先アドレスが設定されることを特徴とするマルチプロセッサシステム。
【請求項６】
グラフィックプロセッサとメインプロセッサを含み、
グラフィックプロセッサは、
メモリからディスプレイリストとして記述される命令を順次読み出すＤＭＡＣと、
読み出された命令を順次解読するデコーダと、
解読された命令がディスプレイリストのメインルーチンに含まれる所定のコール命令であるときメインプロセッサに対して移行用割込を発生し、かつ、解読された命令が前記コール命令によって呼び出されるサブルーチンに含まれるリターン命令であるときメインプロセッサに対して復帰用割込を発生する割込発生部とを備え、
メインプロセッサは、
割込発生部から移行用割込を受けたとき、前記サブルーチンの処理が完了したときメインルーチンへ戻るための戻りアドレスを所定のメモリへ退避するアドレス退避部と、
割込発生部から復帰用割込を受けたとき、前記戻りアドレスを前記所定のメモリから読み出してグラフィックプロセッサへ通知するアドレス通知部とを備え、

グラフィックプロセッサへ通知された戻りアドレスがＤＭＡＣのアクセスの対象アドレスとして設定されることを特徴とするマルチプロセッサシステム。
【請求項７】
グラフィックプロセッサとメインプロセッサを含み、
グラフィックプロセッサは、
メモリからディスプレイリストとして記述される命令を順次読み出すＤＭＡＣと、
読み出された命令を順次解読するデコーダと、
解読された命令がディスプレイリストに含まれる所定のコール命令またはジャンプ命令であるときメインプロセッサに対して割込を発生する割込発生部とを備え、
メインプロセッサは、割込発生部から割込を受けたとき、前記コール命令またはジャンプ命令と、その後に置かれる実行停止命令のフォーマット中に分割して格納されたコール命令またはジャンプ先アドレスを取得してグラフィックプロセッサへ通知するアドレス通知部を備え、
グラフィックプロセッサへ通知されたコール先アドレスまたはジャンプ先アドレスがＤＭＡＣのアクセスの対象アドレスとして設定されることを特徴とするマルチプロセッサシステム。
【請求項８】
第１のプロセッサが、実行中のメインルーチンにおいてコール命令を実行したとき、そのコール命令によって呼び出されるサブルーチンの処理が完了したときメインルーチンへ戻るための戻りアドレスの退避を第２のプロセッサに委託することを特徴とするマルチプロセッサシステムにおけるプログラム実行方法。
【請求項９】
請求項８に記載の方法において、
第１のプロセッサ内部のスタック領域に空きがある場合は第１のプロセッサが自らそのスタック領域へ戻りアドレスを退避する一方、
前記スタック領域に空きがない場合は第２のプロセッサに戻りアドレスの退避を委託することを特徴とするマルチプロセッサシステムにおけるプログラム実行方法。
【請求項１０】
請求項８に記載の方法において、
前記コール命令が、戻りアドレスの退避を第２のプロセッサへ委託する旨を明示しない命令であるとき、第１のプロセッサは、自身内蔵するスタック領域へ戻りアドレスを退避する一方、
前記コール命令が、戻りアドレスの退避を第２のプロセッサへ委託する旨を明示する命令であるとき、第１のプロセッサは、第２のプロセッサに戻りアドレスの退避を委託することを特徴とするマルチプロセッサシステムにおけるプログラム実行方法。
【請求項１１】
第１のプロセッサがコール命令またはジャンプ命令を実行したとき、コール先アドレスまたはジャンプ先アドレスのフルアドレスの取得を第２のプロセッサに委託することを特徴とするマルチプロセッサシステムにおけるプログラム実行方法。
【請求項１２】
請求項１１に記載の方法において、
コール先アドレスまたはジャンプ先アドレスのビット数が第１のプロセッサによって取得可能なビット数を超えているとき、コール先アドレスまたはジャンプ先アドレスのフルアドレスの取得を第２のプロセッサに委託することを特徴とするマルチプロセッサシステムにおけるプログラム実行方法。
【請求項１３】
請求項１１に記載の方法において、
前記コール命令またはジャンプ命令が、コール先アドレスまたはジャンプ先アドレスの取得を第２のプロセッサへ委託する旨を明示する命令であるとき、コール先アドレスまたはジャンプ先アドレスのフルアドレスの取得を第２のプロセッサに委託することを特徴と

するマルチプロセッサシステムにおけるプログラム実行方法。

【書類名】明細書
【発明の名称】マルチプロセッサシステムとそのシステムにおけるプログラム実行方法
【技術分野】
【０００１】
本発明は、コンピュータのプロセッサにマルチプロセッサ構成を用いたシステムと、そのシステムにおいてプログラムを実行する方法に関する。
【背景技術】
【０００２】
リアルタイムマルチメディアアプリケーションの重要性は、より一層高まってきている。こうしたアプリケーションは、マルチプロセッサシステムで処理されることが多くなってきている。例えば、ゲーム機など高速かつ高精細な画像表示が要求される装置では、グラフィックプロセッサ（ＧＰＵ；Graphics Processing Unit）とメインプロセッサとが連携して、高速性が要求されるグラフィック処理を実行する。
【０００３】
一方、ゲーム機、携帯電話、個人情報端末（ＰＤＡ；Personal Digital Assistance）など、携帯型の電子装置は、ますます小型化する傾向にあり、プロセッサやメモリなどを実装する制御回路の小面積化、高集積化が要請される。
【発明の開示】
【発明が解決しようとする課題】
【０００４】
制御回路が小面積化、高集積化されてくると、プログラムの実行などを円滑に行うために実装されるレジスタなどのハードウェア資源を実装する面積も縮小していく。このようなハードウェア資源を少なくしていくと、コマンドによっては、円滑に実行することが難しくなる場合も生じてくる。
【０００５】
本発明は、このような状況に鑑みてなされたものであり、その目的は、小規模な回路で柔軟に処理を実行することができるマルチプロセッサシステムとそのシステムにおけるプログラム実行方法を提供することにある。
【課題を解決するための手段】
【０００６】
上記課題を解決するために、本発明のある態様は、マルチプロセッサシステムにおけるプログラム実行方法である。この方法は、第１のプロセッサが、実行中のメインルーチンにおいてコール命令を実行したとき、そのコール命令によって呼び出されるサブルーチンの処理が完了したときメインルーチンへ戻るための戻りアドレスの退避を第２のプロセッサに委託する。
【０００７】
本発明のさらに別の態様もまた、マルチプロセッサシステムにおけるプログラム実行方法である。この方法は、第１のプロセッサがコール命令またはジャンプ命令を実行したとき、コール先アドレスまたはジャンプ先アドレスのフルアドレスの取得を第２のプロセッサに委託する。
【０００８】
なお、以上の構成要素の任意の組合せ、本発明の表現を方法、装置、システム、記録媒体、コンピュータプログラムなどの間で変換したものもまた、本発明の態様として有効である。
【発明の効果】
【０００９】
本発明によれば、小規模な回路で柔軟に処理を実行することができる。
【発明を実施するための最良の形態】
【００１０】
まず、実施形態を詳細に説明する前に、概要を以下に示す。

ある態様は、第１のプロセッサと第２のプロセッサを含む。第１のプロセッサは、実行中のメインルーチンにおいて所定のコール命令を実行したとき第２のプロセッサに対して割込を発生する割込発生部を備える。第２のプロセッサは、当該割込発生部から割込を受けたとき、コール命令によって呼び出されるサブルーチンの処理が完了したときメインルーチンへ戻るための戻りアドレスを所定のメモリ領域に退避するアドレス退避部を備える。ここで、「メインルーチン」および「サブルーチン」とは、コール元のルーチンとコール先のルーチンとの関係を示した記載表現であり、例えば、特定のルーチンがコール元のルーチンとの関係では「サブルーチン」と、コール先のルーチンとの関係では「メインルーチン」と表現されてもよい。

【００１１】

上記割込発生部は、サブルーチンにおいて所定のリターン命令を実行したとき第２のプロセッサに対して再度割込を発生し、第２のプロセッサは、再度発生された割込を受けたとき、戻りアドレスを第１のプロセッサへ通知するアドレス通知部をさらに備えてもよい。第１のプロセッサは、命令をフェッチするフェッチャーを備え、このフェッチャーによるアクセスの対象アドレスとして戻りアドレスが設定されてもよい。

【００１２】

別の態様も、第１のプロセッサと第２のプロセッサを含む。第１のプロセッサは、所定のコール命令またはジャンプ命令を実行したとき第２のプロセッサに対して割込を発生する割込発生部を備える。第２のプロセッサは、第１のプロセッサから割込を受けたとき、コール命令またはジャンプ命令と、その後に置かれる実行停止命令のフォーマット中に分割して格納されたコール先アドレスまたはジャンプ先アドレスを抽出するアドレス抽出部と、取得したコール先アドレスまたはジャンプ先アドレスを第１のプロセッサに通知するアドレス通知部とを備える。ここで、「ジャンプ命令」は、条件付ジャンプ命令、無条件ジャンプ命令のいずれであってもよい。第１のプロセッサは、命令をフェッチするフェッチャーを備え、このフェッチャーによるアクセスの対象アドレスとしてコール先アドレスまたはジャンプ先アドレスが設定されてもよい。

【００１３】

別の態様は、グラフィックプロセッサとメインプロセッサを含む。グラフィックプロセッサは、メモリからディスプレイリストとして記述される命令を順次読み出すＤＭＡＣ（Direct Memory Access Controller）と、読み出された命令を順次解読するデコーダと、解読された命令がディスプレイリストのメインルーチンに含まれる所定のコール命令であるときメインプロセッサに対して移行用割込を発生し、かつ、解読された命令がコール命令によって呼び出されるサブルーチンに含まれるリターン命令であるときメインプロセッサに対して復帰用割込を発生する割込発生部とを備える。メインプロセッサは、割込発生部から移行用割込を受けたとき、サブルーチンの処理が完了したときメインルーチンへ戻るための戻りアドレスを所定のメモリへ退避するアドレス退避部と、割込発生部から復帰用割込を受けたとき、戻りアドレスを所定のメモリから読み出してグラフィックプロセッサへ通知するアドレス通知部とを備える。グラフィックプロセッサへ通知された戻りアドレスがＤＭＡＣのアクセスの対象アドレスとして設定される。

【００１４】

別の態様も、グラフィックプロセッサとメインプロセッサを含む。グラフィックプロセッサは、メモリからディスプレイリストとして記述される命令を順次読み出すＤＭＡＣと、読み出された命令を順次解読するデコーダと、解読された命令がディスプレイリストに含まれる所定のコール命令またはジャンプ命令であるときメインプロセッサに対して割込を発生する割込発生部とを備える。メインプロセッサは、割込発生部から割込を受けたとき、コール命令またはジャンプ命令と、その後に置かれる実行停止命令のフォーマット中に分割して格納されたコール先アドレスまたはジャンプ先アドレスを取得してグラフィックプロセッサへ通知するアドレス通知部を備える。グラフィックプロセッサへ通知されたコール先アドレスまたはジャンプ先アドレスがＤＭＡＣのアクセスの対象アドレスとして設定される。

【００１５】
第１のプロセッサ内部のスタック領域に空きがある場合は第１のプロセッサが自らそのスタック領域へ戻りアドレスを退避してもよい。一方、スタック領域に空きがない場合は第２のプロセッサに戻りアドレスの退避を委託してもよい。コール命令が、戻りアドレスの退避を第２のプロセッサへ委託する旨を明示しない命令であるとき、第１のプロセッサは、自身内蔵するスタック領域へ戻りアドレスを退避してもよい。一方、コール命令が、戻りアドレスの退避を第２のプロセッサへ委託する旨を明示する命令であるとき、第１のプロセッサは、第２のプロセッサに戻りアドレスの退避を委託してもよい。
【００１６】
コール先アドレスまたはジャンプ先アドレスのビット数が第１のプロセッサによって取得可能なビット数を超えているとき、コール先アドレスまたはジャンプ先アドレスのフルアドレスの取得を第２のプロセッサに委託してもよい。コール命令またはジャンプ命令が、コール先アドレスまたはジャンプ先アドレスの取得を第２のプロセッサへ委託する旨を明示する命令であるとき、コール先アドレスまたはジャンプ先アドレスのフルアドレスの取得を第２のプロセッサに委託してもよい。
【００１７】
以下、本発明の実施形態を詳細に説明する。
図１は、本発明の実施形態におけるマルチプロセッサシステム１００の構成を示す図である。本実施形態におけるマルチプロセッサシステム１００は、同一バス３２上にＧＰＵ１２、ＣＰＵ１４、およびメインメモリ１６が接続された構成である。なお図１のブロック図では、ＧＰＵ１２、ＣＰＵ１４、およびメインメモリ１６に関し、本実施形態の説明に必要な範囲で、それぞれの機能やデータ構造を記載しており、それらの一般的な機能やデータ構造は省略してある。
【００１８】
ＧＰＵ１２は、主に、３次元グラフィック表示に必要な計算やレンダリングを行うグラフィック処理用のプロセッサであり、ワンチップ化された半導体集積回路で構成されてもよい。ＧＰＵ１２は、ＤＭＡＣ２０、デコーダ２２、および割込発生部２４を備える。ＤＭＡＣ２０は、メインメモリ１６との間で直接データ転送を行う。本実施形態では、メインメモリ１６からコマンドをフェッチするフェッチャーとして機能する。
【００１９】
ＤＭＡＣ２０は、アドレススタック２６、アドレスカウンタ２８、およびメモリ制御部３０を含む。アドレススタック２６は、ＧＰＵ１２が現在実行中の描画処理を割込の発生により一時中断し、コールバック関数などを実行する場合、当該コールバック関数に係るサブルーチンを終了し、上記描画処理を再開する際に戻るべき戻りアドレスを格納する。本実施形態では、ＧＰＵ１２の小規模化の要請により、小さな領域とする。例えば、レジスタを２個設け、２つの戻りアドレスを格納可能な設計にしてもよい。この設計では、２回コール命令が実行され、２段のサブルーチンが構築された場合も、このアドレススタック２６に２つの戻りアドレスを格納することができる。それ以上のコール命令が実行された場合、その他の領域に戻りアドレスを格納する必要が生じるが、その領域の確保する手法については後述する。アドレスカウンタ２８は、アクセス先のアドレス、本実施形態ではメインメモリ１６のアドレスを保持し、データ読み出しタイミングにしたがい、そのアドレスをインクリメントしていく。メモリ制御部３０は、メインメモリ１６との間のデータ転送を制御し、具体的には、アドレスカウンタ２８の指示するアドレスからデータを読み出す。
【００２０】
デコーダ２２は、ＤＭＡＣ２０にフェッチされたコマンドを順次解読する。その解読にしたがい、図示しない加算器などを備える実行ユニットは、コマンドを実行し、図示しないフレームメモリなどに演算結果を書き出す。デコーダ２２は、解読したコマンドが、割込を発生させて、後述するディスプレイリスト５０のコール命令であった場合、割込ハンドラ５２を読み出す。割込ハンドラ５２が実行されると、デコーダ２２は、現在実行中の

描画処理を一時中断して、その処理に復帰するための戻りアドレスをアドレススタック２６に格納する。なお、描画処理を一時中断しない場合は、戻りアドレスを格納しなくてもよい。また、アドレススタック２６に空き領域がない場合、割込発生部２４に指示して、ＣＰＵ１４に戻りアドレスの格納を依頼する。

【００２１】

戻りアドレスの格納と共に、デコーダ２２は、初期設定によりフックされているコールバック関数を読み出す。例えば、解読したコマンドフォーマットの下位１６ビットに設定された引数を用いて、当該コールバック関数を呼び出す。この際、現在実行中の描画処理を一時中断させてもよいし、現在実行中の描画処理を継続してもよい。また、描画処理を一時中断した場合、コールバック処理から戻ってきた際、その中断していた処理を再開してもよいし、描画処理の再開を指示する別のコマンドを解読した際に、その処理を再開してもよい。これらは、予めコマンドフォーマットに、コマンドタイプとして設定することができる。また、ホルト命令などの実行停止命令の有無によって、設定してもよい。

【００２２】

また、デコーダ２２は、解読したコマンドが割込を発生させて、後述するディスプレイリスト５０のコール命令またはジャンプ命令を発生させるものであり、かつ、アクセスすべき対象アドレスの一部しか設定されていないタイプであった場合、デコーダ２２が割込発生部２４に指示して、ＣＰＵ１４にディスプレイリスト５０の対象アドレスの抽出を依頼する。なお、コール命令であった場合、上記と同様に戻りアドレスを格納する。本実施形態では、この対象アドレスを、１つのコマンドフォーマット中に設定しきれない場合、複数コマンドに跨って設定する。例えば、ディスプレイリスト５０に割込を発生させるためのコール命令やジャンプ命令のコマンドと、その後にホルト命令やエンドマーカが設定されたコマンドとがセットで記述されるコマンド体系を用いている場合、それら２つのコマンドに分割して設定することができる。そして、上記対象アドレスが３２ビットで指定すべきものであり、１つのコマンドにアドレスを設定可能な領域が２４ビット以下の場合、上記対象アドレスを１６ビットずつに分割して、２つのコマンドフォーマット中に設定されてもよい。

【００２３】

なお、当該アドレスが１つのコマンドに設定可能なデータ長であり、１つのコマンドに設定されている場合、デコーダ２２は、アドレスの抽出をＣＰＵ１４に依頼しない。通常通り、デコーダ２２は、そのアドレスをアドレスカウンタ２８に設定すればよい。対象アドレスが複数のコマンドに分割されて設定されているか否かは、コマンドタイプとして、予めコマンドのフォーマット中に設定しておき、それを解読して判断してもよいし、実行停止コマンドの有無で判断してもよい。

【００２４】

また、デコーダ２２は、解読したコマンドが割込を発生させて、上記コール命令に対応したリターン命令を発生させてるものであり、そのコール命令に対応する戻りアドレスをアドレススタック２６または後述するメインメモリ１６の補助スタック領域５４から読み出す。その戻りアドレスが補助スタック領域５４に格納されいている場合、割込発生部２４に指示して、ＣＰＵ１４に戻りアドレスの読み出しを依頼する。

【００２５】

割込発生部２４の以上の動作をまとめると、以下のイベントが発生した場合、ＣＰＵ１４に割込信号を発生させる。それは、戻りアドレスの格納をＣＰＵ１４に依頼する場合である。また、複数のコマンドに分割して設定されたアドレスの抽出をＣＰＵ１４に依頼する場合である。さらに、サブルーチンにおいて、コール元に戻るためのリターン命令が実行された際、戻りアドレスがメインメモリ１６の補助スタック領域５４に格納されている場合、その戻りアドレスの転送をＣＰＵ１４に依頼する場合である。以下、本明細書中では、ディスプレイリスト５０のメインルーチンに含まれる所定のコール命令を解読、実行したとき、ＣＰＵ１４に発生させる割込信号を移行用割込と呼ぶ。また、コール命令によって呼び出されたサブルーチンに含まれるリターン命令を解読、実行したとき、ＣＰＵ１

４に発生させる割込信号を復帰用割込と呼ぶ。なお、割込を発生させるイベントには、キーボードからの入力など、種々存在するが、ここでは、本実施形態に関連するＧＰＵ１２における描画処理実行中に、コール命令やジャンプ命令を解読、実行した際に発生する割込に限定して説明している。

【００２６】

ＣＰＵ１４は、割込制御部４０、アドレス抽出部４２、アドレス退避部４４、アドレス読出部４６、およびアドレス通知部４８を備える。これらは、後述する割込ハンドラ５２の機能を実現する機能ブロックを示す。割込制御部４０は、ＧＰＵ１２の割込発生部２４から移行用割込および復帰用割込を受け付ける。アドレス抽出部４２は、移行用割込により、複数のコマンドに跨る対象アドレスの抽出を依頼されたとき、それぞれのコマンドからアドレスの一部を抽出する。例えば、コール命令またはジャンプ命令などのコマンド、およびその後に置かれるホルト命令などのコマンドのフォーマット中に分離して設定されたアドレスを抽出する。この対象アドレスは、メインメモリ１６内のコール先アドレスまたはジャンプ先アドレスなどを指す。

【００２７】

ＣＰＵ１４は、ＧＰＵ１２より図示しないレジスタ群を多く備える。シフトレジスタを備えてもよい。例えば、１つのコマンドフォーマットが３２ビット仕様で、アドレスの設定可能ビット数が２４ビットまでと規定される場合、２つのコマンドのそれぞれの下位１６ビットを利用して、３２ビットで規定されるアドレスを１６ビットずつに分割して、２つのコマンドに設定する。そして、当該アドレスの上位１６ビットを先のコマンドに設定し、その下位１６ビットを後のコマンドに設定する。アドレス抽出部４２は、先のコマンドからアドレスの上位１６ビットを抽出し、シフトレジスタに入れる。そのシフトレジスタでＭＳＢ（Most Significant Bit）側に１６桁分シフトさせ、それと共に、後のコマンドからアドレスの下位１６ビットを抽出し、シフトレジスタ内のデータと組み合わせることにより、分割前のアドレスに復元することができる。

【００２８】

アドレス通知部４８は、アドレス抽出部４２が生成した対象アドレスをＧＰＵ１２に転送し、アドレスカウンタ２８にそれを設定する。アドレスカウンタ２８は、この処理により保持データが更新され、以降、この設定されたアドレスからインクリメントしていく。

【００２９】

アドレス退避部４４は、移行用割込により、上記戻りアドレスの格納を依頼された場合、後述するメインメモリ１６内の補助スタック領域５４に、その戻りアドレスを格納する。複数の戻りアドレスを格納する場合、ＬＩＦＯ（Last-In First-Out）方式で補助スタック領域５４に格納していくとよい。

【００３０】

アドレス読出部４６は、復帰用割込により、戻りアドレスの転送を依頼された場合、メインメモリ１６内の補助スタック領域５４から、その戻りアドレスを読み出す。アドレス読出部４６は、補助スタック領域５４がＬＩＦＯ方式で制御されている場合、最後に格納された戻りアドレスを読み出す。この読み出す順番は、ＧＰＵ１２からの戻りアドレス転送の依頼順にも合致する。

【００３１】

アドレス通知部４８は、アドレス読出部４６が読み出した戻りアドレスをＧＰＵ１２に転送し、アドレスカウンタ２８にそれを設定する。アドレスカウンタ２８は、この処理により保持データが更新され、以降、この設定されたアドレスからインクリメントしていく。

【００３２】

メインメモリ１６は、各種のコマンドやデータを格納する。本実施形態では、主に、ディスプレイリスト５０、割込ハンドラ５２を格納する。その他、テクスチャデータなどが格納されてもよい。また、メインメモリ１６には、補助スタック領域５４が確保される。ディスプレイリスト５０とは、描画実行時に効率的に処理できるよう、描画に関するコマ

ンド群をひとまとめにリスト化したものである。本実施形態では、ＧＰＵ１２にフェッチされて実行される。割込ハンドラ５２は、割込を処理、制御するために待機しているプログラムである。本実施形態では、主に、ディスプレイリスト５０の実行中にコール命令やジャンプ命令が実行される場合の、ＧＰＵ１２およびＣＰＵ１４の動作を制御するものである。

【００３３】

補助スタック領域５４は、本実施形態では、上記戻りアドレスを格納するために確保される領域である。例えば、メインメモリ１６の最終の領域から先頭アドレスに戻る方向に、転送されてきた戻りアドレスを順々に格納していく。例えば、コール命令により実行されるサブルーチンの段数に対応して、補助スタック領域５４を確保しておけばよい。

【００３４】

図２は、ディスプレイリスト５０の一例を示す図である。ＤＭＡＣ２０は、メインメモリ内のディスプレイリスト５０の先頭アドレスから順番にフェッチしていく。図２において、コマンド”cmd”は、描画コマンドを総称したものであり、プリミティブを規定するためのワイヤー・フレームやポリゴンの指定、色の指定などを行うコマンドが該当する。

【００３５】

ＧＰＵ１２は、デコーダ２２がアドレス＃１に格納されたコマンド”call id”を解読すると、引数iDを利用して、予めフックしてあるコールバックス関数を呼び出す（処理ｓ）。ここで、＃記号は１６進数表記を示す。図２では、予めアドレス＃１０００にコールバック関数が設定されている。そして、アドレス＃１０００から始まるサブルーチンを実行していき、デコーダ２２がその終了を示すコマンド”return”を解読すると、戻りアドレス＃２に移行し、コール元のメインルーチンへ復帰する（処理ｔ）。なお、この戻りアドレスは、アドレススタック２６またはメインメモリ１６内の補助スタック領域５４に格納されている。上記サブルーチンをコールした際に格納されたものである。

【００３６】

次に、ＧＰＵ１２は、デコーダ２２がアドレス＃１００に格納されたコマンド”cpu call #20”を解読すると、そのコマンドと次のアドレス＃１０１に格納されたコマンド”halt #FF”に分割されて設定されているアドレス＃２０ＦＦをＣＰＵ１４に抽出してもらい、そのアドレス＃２０ＦＦから始まるサブルーチンを呼び出す（処理ｕ）。そして、そのアドレス＃２０ＦＦから始まるサブルーチンを実行していき、デコーダ２２がその終了を示すコマンド”return”およびそれとセットのコマンド”halt”を解読すると、戻りアドレス＃１０２に移行し、メインルーチンに復帰する（処理ｖ）。

【００３７】

次に、ＧＰＵ１２は、デコーダ２２がアドレス＃１に格納されたコマンド”jmp #220”を解読すると、アドレス＃２２０にジャンプする（ｗ）。この場合、同一ルーチン内の分岐などによるジャンプであり、サブルーチンを呼び出し、その終了後にコール元に復帰させるという処理ではないため、戻りアドレスの格納は必要ない。ここでは、アドレスが２４ビット以内で規定できる場合、複数のコマンドに分割して設定する必要はなく、１つのコマンドでアドレスを設定することができる。この処理、一般のジャンプ命令の処理と同様である。

【００３８】

次に、ＧＰＵ１２は、デコーダ２２がアドレス＃３００に格納されたコマンド”cpu jmp #40”を解読すると、そのコマンドと次のアドレス＃３０１に格納されたコマンド”halt #AA”に分割されて設定されているアドレス＃４０ＡＡをＣＰＵ１４に抽出してもらい、そのアドレス＃４０ＡＡにジャンプする（処理ｘ）。なお、コマンド中の”cpu”は、ＣＰＵ１４を介在させる処理であることを示す。

【００３９】

図３は、本実施形態におけるＧＰＵ１２がＣＰＵ１４にアドレス抽出を指示する第１コマンド例のフォーマットを示す。コマンド”cpu call”は、”halt”とセットで使用され、ディスプレイリスト５０のコール命令を実行するものである。図３では、それぞれのコマ

ンドが３２ビットで規定され、上記８ビットすなわち３１～２４桁にコマンドコードが、そこから下位に８ビットすなわち２３～１６桁にコマンドタイプが、下位１６ビットすなわち１５～０桁にデータフィールドとして、コール先アドレスの上位または下位１６ビット分が記述されている。ここで、コマンドコードは、各コマンドをコード化したものである。コマンドタイプは、割込を発生させた場合に実行中の処理を中断するか否か、戻りアドレスの保存が必要か否か、ＧＰＵ１２単独の処理か否か、換言すればＣＰＵ１４と協働する処理か否か、といったコマンドの実行に関する付随的情報を指定するものである。なお、これらのフォーマットは一例であり、デコーダ２２が正確にコマンドを解読できれば、任意にフィールドを設定し、命令やデータを記述することができる。例えば、上位８ビットでコマンドの命令部を記述できる場合、残りの２４ビットをデータフィールドとして、そこにアドレスや引数を記述してもよい。

【００４０】

図４は、本実施形態におけるＧＰＵ１２がＣＰＵ１４にアドレス抽出を指示する第２コマンド例のフォーマットを示す。コマンド”cpu jmp”は、”halt”とセットで使用され、ディスプレイリスト５０のジャンプ命令を実行するものである。このジャンプ命令には、分岐によるジャンプ命令や無条件のジャンプ命令などが含まれる。図４も、図３と同様に、それぞれのコマンドの下位１６ビット分のフィールドに、ジャンプ先アドレスの上位または下位１６ビット分が記述されている。

【００４１】

図５は、本実施形態におけるコール命令を実行した際のフローチャートである。ＧＰＵ１２は、デコーダ２２がコマンド”cpu call”を解読すると、そのコマンドを実行する（Ｓ１０）。次に、デコーダ２２がその次のコマンド”halt”を解読し、現在実行中の処理を一時中断する（Ｓ１２）。ＧＰＵ１２は、これに平行して、ＣＰＵ１４に対して移行割込を発生させる（Ｓ１４）。

【００４２】

ＣＰＵ１４は、この移行割込に応じて、ＧＰＵ１２が実行していたコール元のメインルーチンに復帰するための戻りアドレスをメインメモリ１６内の補助スタック領域５４に退避させる（Ｓ１６）。それと共に、上記コマンド”cpu call”および コマンド”halt”に分割して格納されているコール先アドレスを抽出する（Ｓ１８）。そして、そのコール先アドレスをＧＰＵ１２に通知する（Ｓ２０）。

【００４３】

ＧＰＵ１２は、コール先アドレスから始まるサブルーチンに移行する（Ｓ２２）。そのサブルーチンを実行していき、デコーダ２２がコマンド”return”を解読すると、そのコマンドを実行する（Ｓ２４）。デコーダ２２がその次のコマンド”halt”を解読し、現在実行中の処理を終了する（Ｓ２６）。ＧＰＵ１２は、これに平行して、ＣＰＵ１４に対して復帰割込を発生させる（Ｓ２８）。

【００４４】

ＣＰＵ１４は、この復帰割込に応じて、補助スタック領域５４に退避させていた戻りアドレスをメインメモリ１６内の補助スタック領域５４から読み出す（Ｓ３０）。そして、その戻りアドレスをＧＰＵ１２に通知する（Ｓ３２）。ＧＰＵ１２は、その戻りアドレスから、コール元のメインルーチンに復帰し、処理を続行する（Ｓ３４）。

【００４５】

なお、以上説明した処理は、ＧＰＵ１２のアドレススタック２６に空き領域がないことが前提の処理である。この点、アドレススタック２６に空き領域がある場合は、戻りアドレスをそこに退避してもよい。また、ＧＰＵ１２内にフラグレジスタなどを設け、アドレススタック２６に空き領域があるか否かを示すフラグをそのレジスタに立ててもよい。この場合、デコーダ２２は、そのフラグを参照して、ＣＰＵ１４に戻りアドレスの退避を依頼するか否かを判断することができる。戻りアドレスをＧＰＵ１２のアドレススタック２６に格納する場合、図５のステップＳ１６、Ｓ２８～Ｓ３２の処理は不要となり、ＧＰＵ１２内での処理となる。

【００４６】
図６は、本実施形態におけるジャンプ命令を実行した際のフローチャートである。ＧＰＵ１２は、デコーダ２２がコマンド”cpu jmp”を解読すると、そのコマンドを実行する（Ｓ５０）。デコーダ２２がその次のコマンド”halt”を解読し、現在実行中の処理を終了する（Ｓ５２）。ＧＰＵ１２は、これに平行して、ＣＰＵ１４に対して割込を発生させる（Ｓ５４）。
【００４７】
ＣＰＵ１４は、この割込に応じて、上記コマンド”cpu jmp”および コマンド”halt”に分割して格納されているジャンプ先アドレスを抽出する（Ｓ５６）。そして、そのジャンプ先アドレスをＧＰＵ１２に通知する（Ｓ５８）。ＧＰＵ１２は、ジャンプ先アドレスに移行し（Ｓ６０）、そのアドレスから読み出しを再開する。
【００４８】
以上説明したように本実施形態によれば、コール命令が割込として発生した場合、割込が発生したコール元のメインルーチンに復帰するための戻りアドレスをＧＰＵ１２外のメインメモリ１６に退避することができることから、ＧＰＵ１２内の戻りアドレスを格納する領域を縮小させることができる。すなわち、コール先のルーチンでさらにコール命令が発生するという処理が複数回実行され、複数段のサブルーチンが構築される場合、コール元のルーチンに復帰するための戻りアドレスを複数、保持する必要がある。これをＧＰＵ１２内にすべて格納しようとすると、それだけレジスタを多く設ける必要が生じ、ＧＰＵ１２の小規模化を難しくする。これに対し、ＧＰＵ１２のレジスタを削減し、小規模化を図ると、複数段のサブルーチンを構築することが難しくなり、処理の柔軟性が低下してしまう。この点、本実施形態では、戻りアドレスをメインメモリ１６に退避することができることから、ＧＰＵ１２の小規模化と処理の柔軟性を両立することができる。具体的には、ハードウェア資源的に簡素なＧＰＵ１２でも、多数段のコール命令を処理することができる。
【００４９】
また、１つのコマンドフィールド中に、コール先アドレスやジャンプ先アドレスが入りきらず、複数のコマンドに渡ってそれらのアドレスが設定される場合、そのアドレスの抽出をＣＰＵ１４に委託することにより、ＧＰＵ１２のレジスタを削減することができる。すなわち、複数のコマンドに跨るアドレスを抽出するには、それぞれのコマンドに含まれるアドレスの一部を抽出し、それをシフトさせたり、合成させたりするためのレジスタが必要となる。そのレジスタは、１つのコマンドに設定された引数やアドレスを抽出する場合に必要なレジスタより大きな規模となる。この点、本実施形態では、複数のコマンドに分割して設定されたアドレスを抽出するなど、シフトや加算などの論理演算が必要なコマンドの解読をメインのＣＰＵ１４に委託する。これにより、ＧＰＵ１２は、１コマンド単位で解読可能なコマンドを処理するに必要なレジスタを備えればよいことになり、回路面積を削減し、小規模化を図ることができる。また、ＧＰＵ１２のハードウェア資源では解読が難しいコマンドは、ＣＰＵ１４が代わりに解読することにより、様々な記述体系のコマンドに対応することもできる。よって、ＧＰＵ１２の小規模化と処理の柔軟性を両立することができる。ＧＰＵ１２が小規模化されれば、マルチプロセッサシステム全体を小規模化することができる。
【００５０】
以上、実施形態をもとに本発明を説明した。なお本発明はこれらの実施形態に限定されることなく、そのさまざまな変形例もまた、本発明の態様として有効である。例えば、実施形態では、ＧＰＵ１２にアドレススタック２６を備えた構成を説明した。この点、ＧＰＵ１２にアドレススタック２６を設けずに、すべての戻りアドレスをメインメモリ１６のスタック領域に格納してもよい。これによれば、さらにＧＰＵ１２を小規模化することができる。
【００５１】
また、実施形態では、コール先アドレスおよびジャンプ先アドレスを２つのコマンドに

分割して設定した。この点、１つのコマンドのデータフィールドが小さい場合や、設定するアドレスのデータ長がさらに長い場合などには、３つ以上のコマンドに分割して設定してもよい。これによれば、コマンドの記述体系を柔軟に規定することができる。また、１つのコマンドフォーマットを短くした場合、さらにＧＰＵ１２のハードウェア資源を小規模化することも可能になる。

【図面の簡単な説明】

【００５２】

【図１】本発明の実施形態におけるマルチプロセッサシステムの構成を示す図である。

【図２】本実施形態におけるディスプレイリストの一例を示す図である。

【図３】本実施形態におけるＧＰＵがＣＰＵにアドレス抽出を指示する第１コマンド例のフォーマットを示す図である。

【図４】本実施形態におけるＧＰＵがＣＰＵにアドレス抽出を指示する第２コマンド例のフォーマットを示す図である。

【図５】本実施形態におけるコール命令を実行した際のフローチャートである。

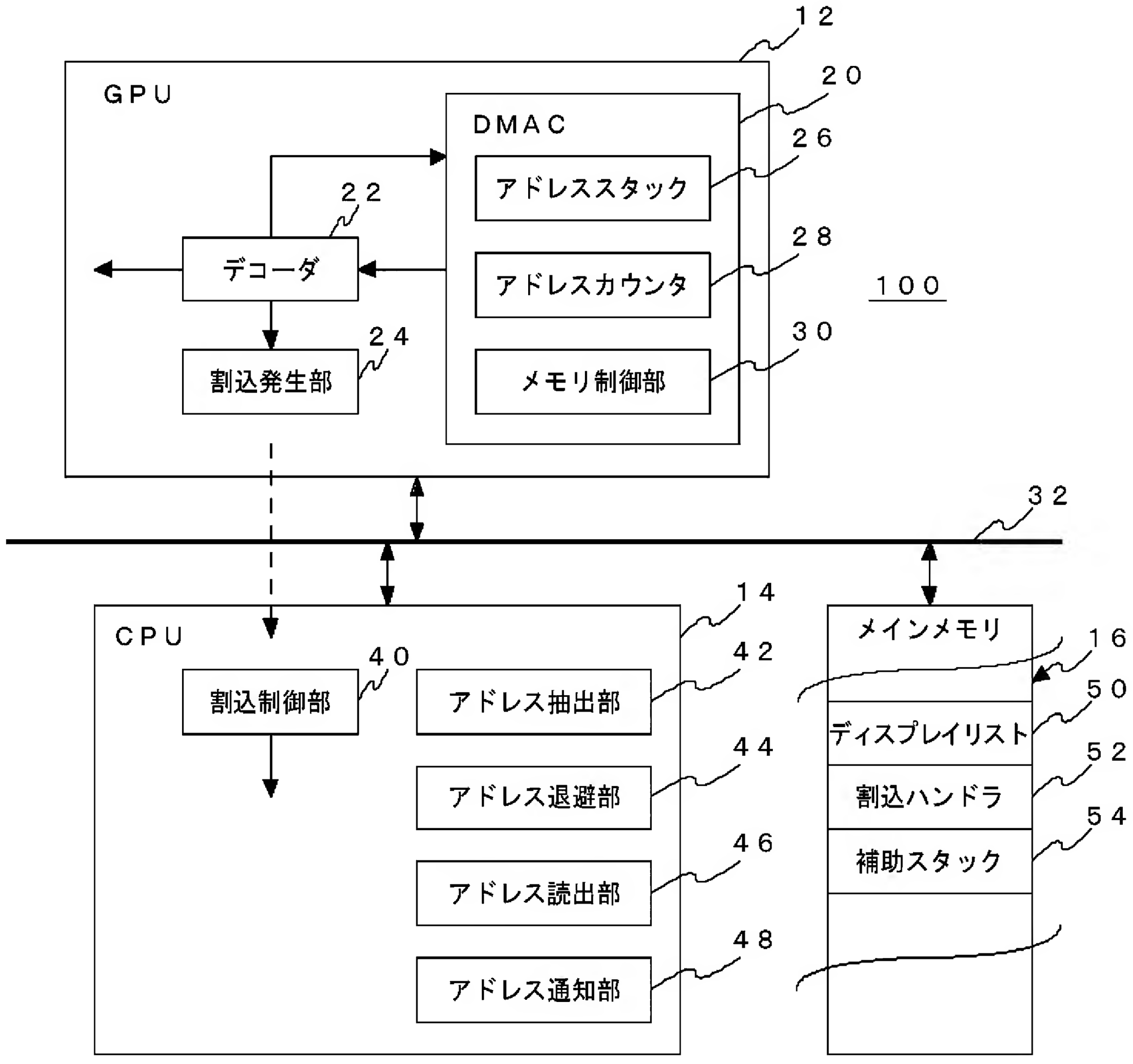
【図６】本実施形態におけるジャンプ命令を実行した際のフローチャートである。

【符号の説明】

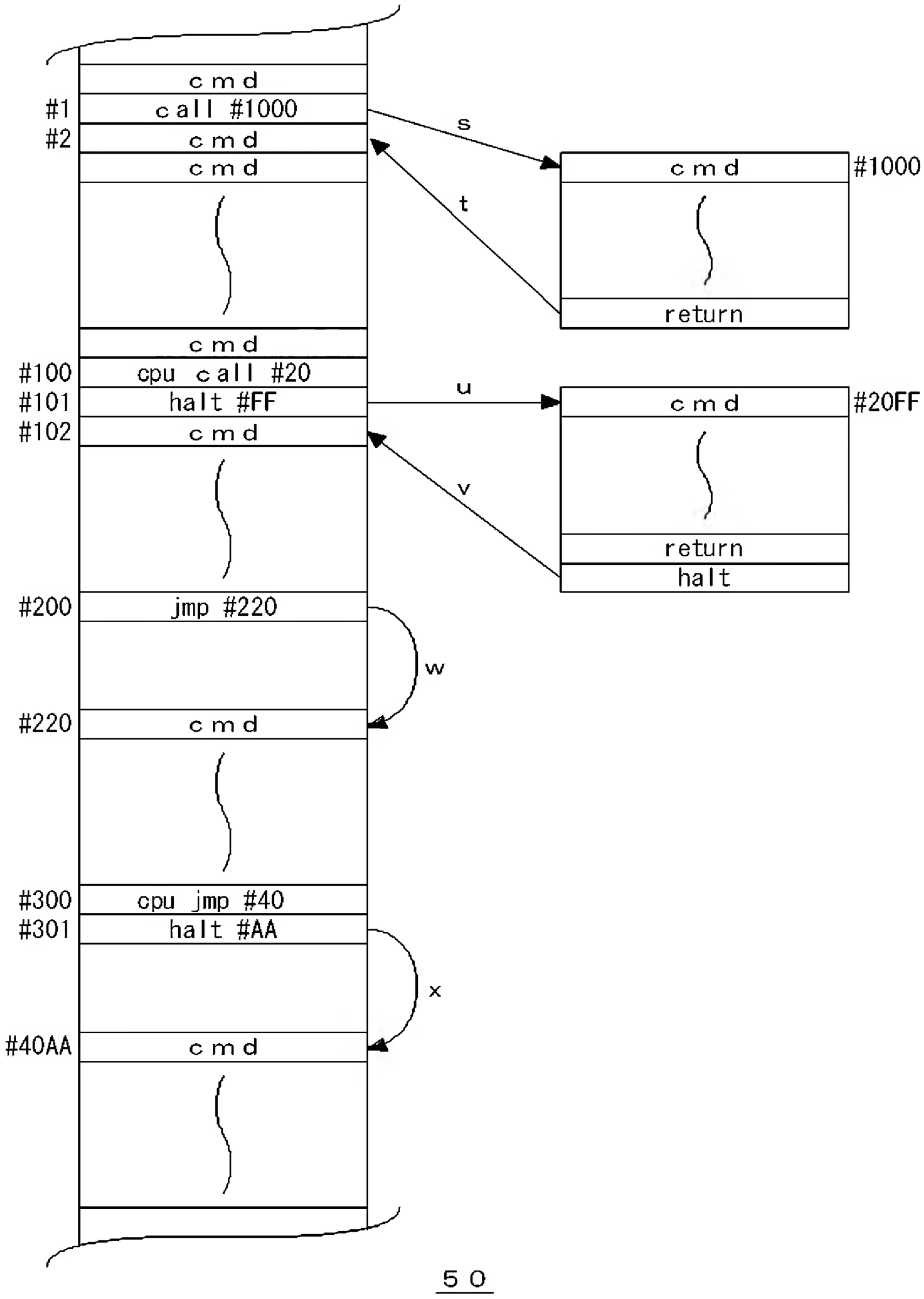
【００５３】

１２　ＧＰＵ、　１４　ＣＰＵ、　１６　メインメモリ、　２０　ＤＭＡＣ、　２２　デコーダ、　２４　割込発生部、　２６　アドレススタック、　２８　アドレスカウンタ、　３０　メモリ制御部、　４０　割込制御部、　４２　アドレス抽出部、　４４　アドレス退避部、　４６　アドレス読出部、　４８　アドレス通知部、　５０　ディスプレイリスト、　５２　割込ハンドラ、　５４　補助スタック領域、　１００　マルチプロセッサシステム。

【書類名】図面
【図１】


12
GPU
20
DMAC
26
アドレススタック
22
デコーダ
28
アドレスカウンタ
100
24
割込発生部
30
メモリ制御部
32
14
CPU
40
割込制御部
42
アドレス抽出部
44
アドレス退避部
46
アドレス読出部
48
アドレス通知部
メインメモリ
16
50
ディスプレイリスト
52
割込ハンドラ
54
補助スタック

【図２】



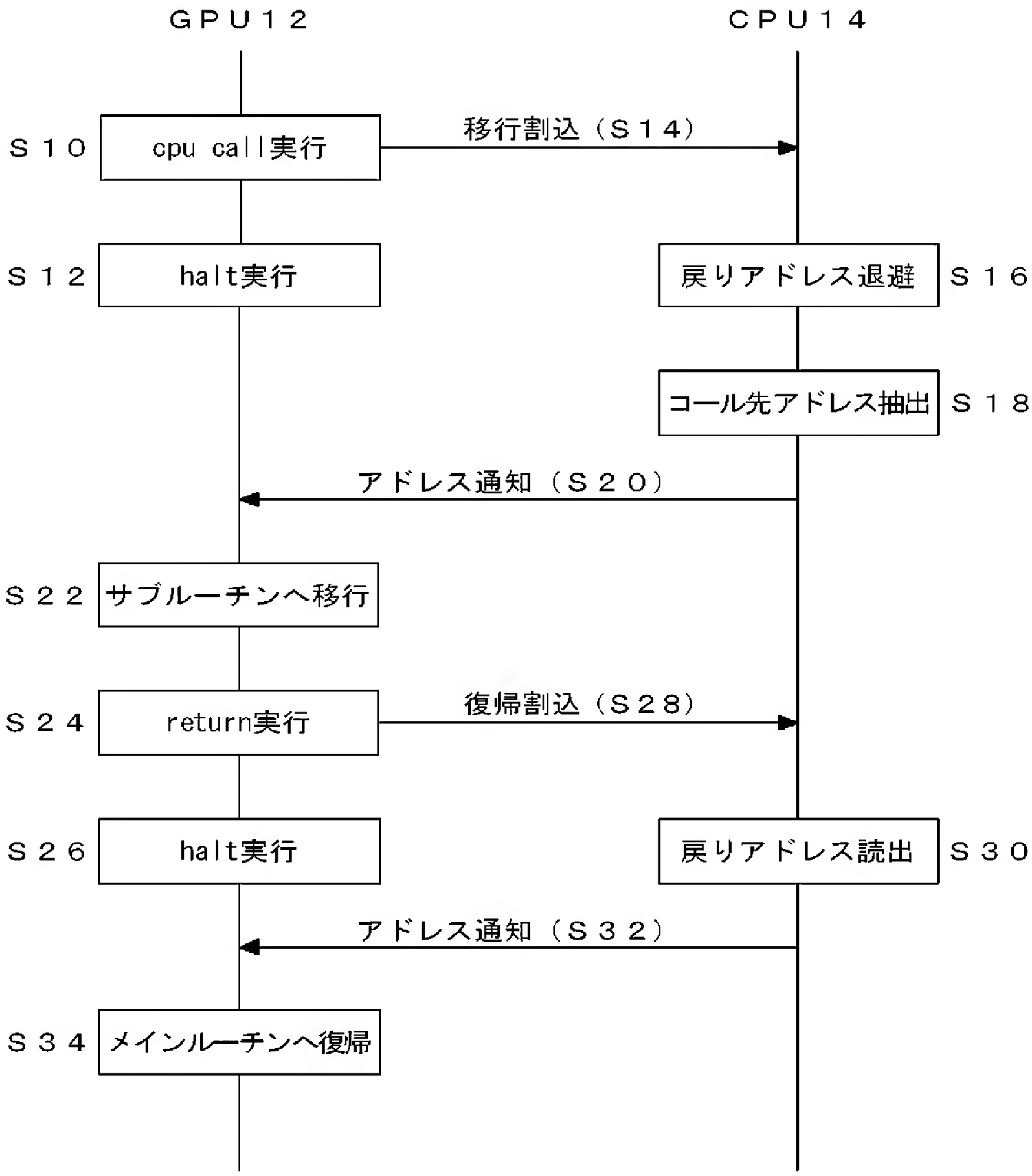
【図３】

	31	23	15	0
cpu call	E 1	O 1	addr 16-31	
halt	E 2	—	addr 0-15	

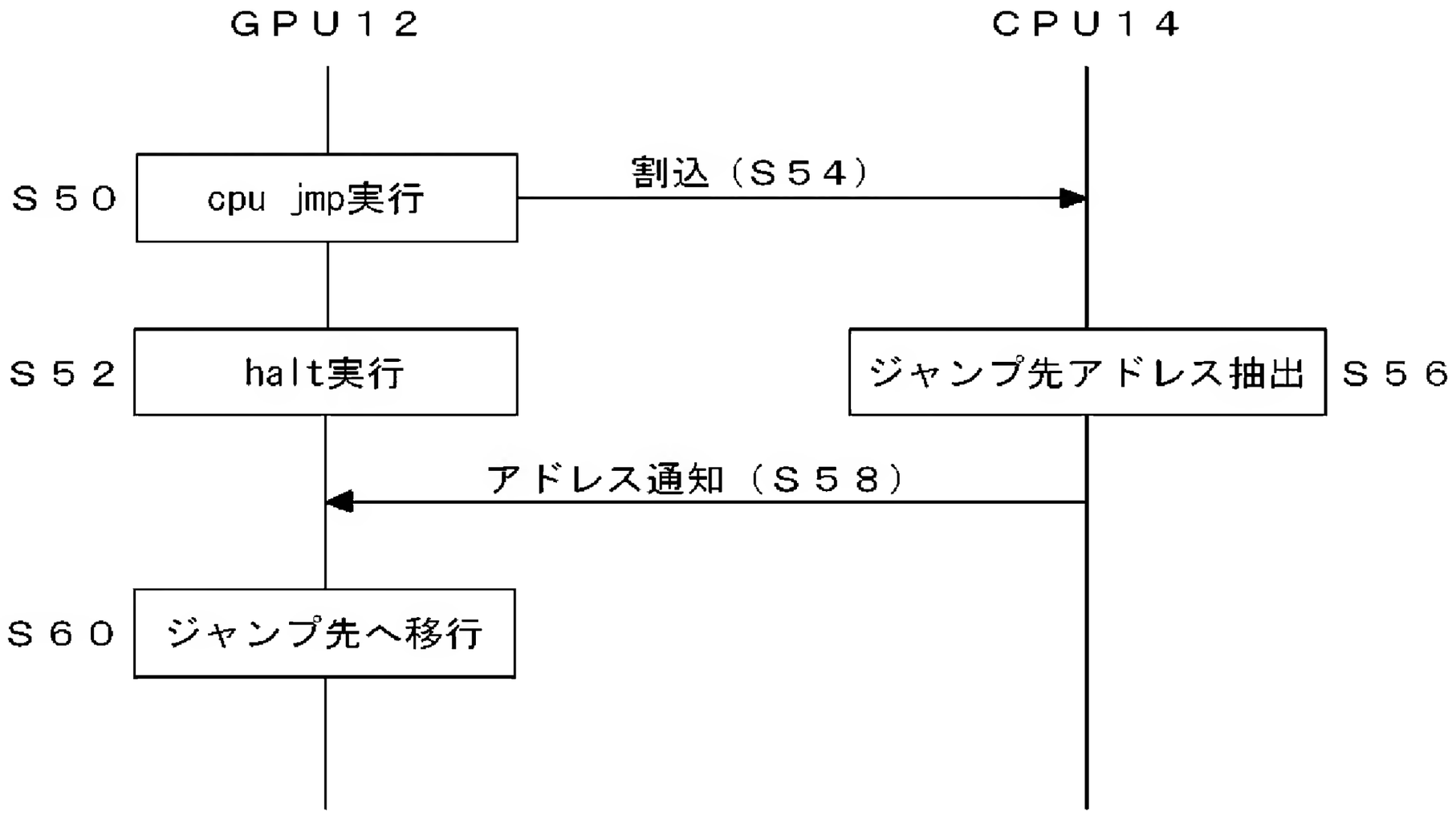
【図４】

	31	23	15	0
cpu jmp	Ｅ１	００	addr 16-31	
halt	Ｅ２	—	addr 0-15	

【図５】



【図６】


ＧＰＵ１２
ＣＰＵ１４
Ｓ５０
cpu jmp実行
割込（Ｓ５４）
Ｓ５２
halt実行
ジャンプ先アドレス抽出
Ｓ５６
アドレス通知（Ｓ５８）
Ｓ６０
ジャンプ先へ移行

【書類名】要約書
【要約】
【課題】　小規模回路で柔軟に処理を実行する。
【解決手段】　マルチプロセッサシステム１００において、第１のプロセッサの割込発生部２４は、実行中のメインルーチンにおいて所定のコール命令を実行したとき第２のプロセッサに対して割込を発生させる。第２のプロセッサのアドレス退避部４４は、割込発生部２４から割込を受けたとき、コール命令によって呼び出されるサブルーチンの処理が完了したときメインルーチンへ戻るための戻りアドレスを所定のメモリ領域に退避する。割込発生部２４は、サブルーチンにおいて所定のリターン命令を実行したとき第２のプロセッサに対して再度割込を発生させ、第２のプロセッサのアドレス通知部４８は、再度発生された割込を受けたとき、戻りアドレスを第１のプロセッサへ通知する。
【選択図】　図１

出願人履歴

３９５０１５３１９
20030701
住所変更

東京都港区南青山二丁目６番２１号
株式会社ソニー・コンピュータエンタテインメント